**EESHANYA JOSHI**

**Batch: A3 Roll No.: 16010122074**

**Experiment / assignment / tutorial No.05**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff Incharge with date**

|  |
| --- |
| **TITLE :Vector** |

**AIM:** Create a class Employee which stores EName, EId and ESalary of an Employee. Use class Vector to maintain an array of Employee with respect to the ESalary. Provide the following functions

1) Create (): this function will accept the n Employee records in any order and will arrange them in the sorted order.

2) Insert (): to insert the given Employee record at appropriate index in the vector depending upon the ESalary.

3) delete ByEname( ): to accept the name of the Employee and delete the record having given name

4) deleteByEId ( ): to accept the Id of the Employee and delete the record having given EId.

Provide the following functions

1. boolean add(E e) : This method appends the specified element to the end of this Vector.
2. void addElement(E obj) This method adds the specified component to the end of this vector, increasing its size by one.
3. int lastIndexOf(Object o, int index) This method returns the index of the last occurrence of the specified element in this vector, searching backwards from index, or returns 1 if the element is not found.
4. void removeElementAt(int index)This method deletes the component at the specified index.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO2:** Explore arrays, vectors, classes and objects in C++ and Java.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Ralph Bravaco , Shai Simoson , “Java Programing From the Group Up” Tata McGrawHill.

2.Grady Booch, Object Oriented Analysis and Design .

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

Vectors in Java are one of the most commonly used data structures. Similar to Arrays data structures which hold the data in a linear fashion. Vectors also store the data in a linear fashion, but unlike Arrays, they do not have a fixed size. Instead, their size can be increased on demand.

Vector class is a child class of AbstractList class and implements on List interface. To use Vectors, we first have to import Vector class from java.util package:

import java.util.Vector;

**Access Elements in Vector:**

We can access the data members simply by using the index of the element, just like we access the elements in Arrays.

Example If we want to access the third element in a vector v, we simply refer to it as v[3].

**Vectors Constructors**

Listed below are the multiple variations of vector [constructors](https://www.edureka.co/blog/constructor-in-java/) available to use:

1. **Vector(int initialCapacity, int Increment)** – Constructs a vector with given initialCapacity and its Increment in size.
2. **Vector(int initialCapacity)*–***Constructs an empty vector with given initialCapacity. In this case, Increment is zero.
3. **Vector()** – Constructs a default vector of capacity 10.
4. **Vector(Collection c)*–***Constructs a vector with a given collection, the order of the elements is same as returned by the collection’s iterator.

There are also three protected parameters in vectors

* + **Int capacityIncrement()** It automatically increases the capacity of the vector when the size becomes greater than capacity.
  + **Int elementCount()** – tell number of elements in the vector
  + **Object[] elementData()** – array in which elements of vector are stored

**Memory allocation of vectors:**

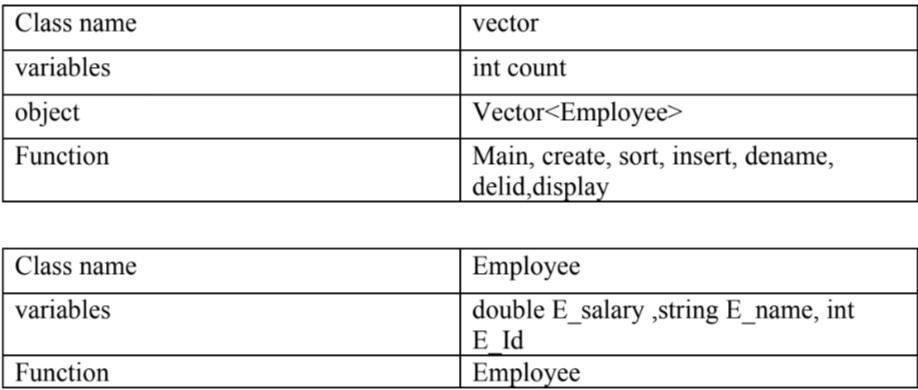
Vectors do not have a fixed size, instead, they have the ability to change their size dynamically. One might think that the vectors allocate indefinite long space to store objects. But this is not the case. Vectors can change their size based on two fields ‘capacity’ and ‘capacityIncrement’. Initially, a size equal to ‘capacity’ field is allocated when a vector is declared. We can insert the elements equal to the capacity. But as soon as the next element is inserted, it increases the size of the array by size ‘capacityIncrement’. Hence, it is able to change its size dynamically.

For a default constructor, the capacity is doubled whenever the capacity is full and a new element is to be inserted.

**Methods of Vectors :**

* Adding elements
* Removing elements
* Changing elements
* Iterating the vector

**Class Diagram:**



**Algorithm:**

**1. Employee Class:**

Create a class Employee with attributes eName, eId, and eSalary, along with a toString() method for display.

**2. EmployeeVector Class:**

Create a class EmployeeVector to manage a list of employees using an ArrayList.

**3. Constructor:**

Initialize an empty ArrayList<Employee> in the constructor.

**4. create() Method:**

* Accept the number of employees (n) as user input.
* Loop n times to gather details for each employee:
* Input eName, eId, and eSalary.
* Create an Employee object.
* Call the insert() method to add the employee to the list.

**5. insert(Employee employee) Method:**

* Find the correct position in the list based on eSalary.
* Use a loop to iterate through existing employees.
* Insert the new employee at the appropriate index to maintain sorted order based on eSalary.

**6. deleteByEName(String eName) Method:**

* Iterate through the list and compare eName with the target name.
* If found, remove the employee and display a success message.

**7. deleteByEId(int eId) Method:**

* Iterate through the list and compare eId with the target ID.
* If found, remove the employee and display a success message.

**8. display() Method:**

Loop through the list and display each employee's details.

**9. Main Method:**

* In the main method, create an instance of EmployeeVector.
* Use a loop to present a menu to the user with options to create records, insert, delete by name/ID, display, or exit.
* Perform the chosen operation based on user input.

**Implementation details:**

import java.util.\*;

class Employee {

    String eName;

    int eId;

    double eSalary;

    Employee(String eName, int eId, double eSalary) {

        this.eName = eName;

        this.eId = eId;

        this.eSalary = eSalary;

    }

    @Override

    public String toString() {

        return "E-Name: " + eName + ", E-Id: " + eId + ", E-Salary: " + eSalary;

    }

}

class EmployeeVector {

    private List<Employee> employeeList;

    public EmployeeVector() {

        employeeList = new ArrayList<>();

    }

    public void create() {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the number of employees: ");

        int n = sc.nextInt();

        sc.nextLine(); // Consume the newline

        for (int i = 0; i < n; i++) {

            System.out.println("Enter details for Employee " + (i + 1) + ":");

            System.out.print("E-Name: ");

            String eName = sc.nextLine();

            System.out.print("E-Id: ");

            int eId = sc.nextInt();

            System.out.print("E-Salary: ");

            double eSalary = sc.nextDouble();

            sc.nextLine(); // Consume the newline

            Employee employee = new Employee(eName, eId, eSalary);

            insert(employee);

        }sc.close();

    }

    public void insert(Employee employee) {

        int index = 0;

        for (Employee emp : employeeList) {

            if (employee.eSalary <= emp.eSalary) {

                break;

            }

            index++;

        }

        employeeList.add(index, employee);

        System.out.println("Employee added successfully.");

    }

    public void deleteByEName(String eName) {

        for (Iterator<Employee> iterator = employeeList.iterator(); iterator.hasNext(); ) {

            Employee emp = iterator.next();

            if (emp.eName.equals(eName)) {

                iterator.remove();

                System.out.println("Employee with E-Name " + eName + " deleted.");

                return;

            }

        }

        System.out.println("Employee with E-Name " + eName + " not found.");

    }

    public void deleteByEId(int eId) {

        for (Iterator<Employee> iterator = employeeList.iterator(); iterator.hasNext(); ) {

            Employee emp = iterator.next();

            if (emp.eId == eId) {

                iterator.remove();

                System.out.println("Employee with E-Id " + eId + " deleted.");

                return;

            }

        }

        System.out.println("Employee with E-Id " + eId + " not found.");

    }

    public void display() {

        for (Employee emp : employeeList) {

            System.out.println(emp);

        }

    }

    public static void main(String[] args) {

        EmployeeVector employeeVector = new EmployeeVector();

        Scanner sc = new Scanner(System.in);

        int choice;

        do {

            System.out.println("\nMenu:");

            System.out.println("1. Create Employee Records");

            System.out.println("2. Insert Employee");

            System.out.println("3. Delete Employee by E-Name");

            System.out.println("4. Delete Employee by E-Id");

            System.out.println("5. Display Employee Records");

            System.out.println("6. Exit");

            System.out.print("Enter your choice: ");

            choice = sc.nextInt();

            sc.nextLine(); // Consume the newline

            switch (choice) {

                case 1:

                    employeeVector.create();

                    break;

                case 2:

                    System.out.println("Enter details for the Employee to insert:");

                    System.out.print("E-Name: ");

                    String eName = sc.nextLine();

                    System.out.print("E-Id: ");

                    int eId = sc.nextInt();

                    System.out.print("E-Salary: ");

                    double eSalary = sc.nextDouble();

                    sc.nextLine(); // Consume the newline

                    Employee employee = new Employee(eName, eId, eSalary);

                    employeeVector.insert(employee);

                    break;

                case 3:

                    System.out.print("Enter E-Name to delete: ");

                    eName = sc.nextLine();

                    employeeVector.deleteByEName(eName);

                    break;

                case 4:

                    System.out.print("Enter E-Id to delete: ");

                    eId = sc.nextInt();

                    sc.nextLine(); // Consume the newline

                    employeeVector.deleteByEId(eId);

                    break;

                case 5:

                    employeeVector.display();

                    break;

                case 6:

                    System.out.println("Exiting...");

                    break;

                default:

                    System.out.println("Invalid choice. Please try again.");

            }

        } while (choice != 6);

        sc.close();

    }

}

**Conclusion:**

In this experiment we were able to understand the basic concepts of vector of objects in java, with the help of which we compiled a program to create a vector of objects which can be used to store Employee info like id, Salary and name, and can add, delete or display n number of Employee info using vector methods.

**Date:\_\_\_\_\_\_\_ Signature of faculty incharge**

**Post Lab Descriptive Questions**

1. **What is the output of the following Program**

|  |
| --- |
| import java.util.;  class demo2 {      public static void main(String[] args)      {          Vector v = new Vector(20);          v.addElement("Geeksforgeeks");          v.insertElementAt("Java", 2);          System.out.println(v.firstElement());      }  } |

**Output: Geeksforgeeks**

1. **Expain any 10 methods of Vector class in detail with the help of example**

* **add()** It is used to append the specified element in the given vector.
* **capacity()**

It is used to get the current capacity of this vector.

* **clear()** It is used to delete all of the elements from this vector.
* **contains()**

It returns true if the vector contains the specified element.

* **indexOf()**

It is used to get the index of the first occurrence of the specified element in the vector. It returns 1 if the vector does not contain the element.

* **insertElementAt()**

It is used to insert the specified object as a component in the given vector at the specified index.

* **elementAt()**

It is used to get the component at the specified index.

* **isEmpty()**

It is used to check if this vector has no components.

* **remove()**

It is used to remove the specified element from the vector. If the vector does not contain the element, it is unchanged.

* **sort()**

It is used to sort the list according to the order induced by the specified Comparator.

**CODE:**

import java.util.;

public class Vectors

{     public static void main(String[] args)

    {

        Vector <Integer> v = new Vector<>(10);

        v.add(100);

        v.add(1689);

        v.add(2675);

        v.add(12557);

        // 1. add()

        System.out.println("\n1. add()");

        System.out.println("\tVector before adding element : "+ v);

        v.add(14);

        //2.capacity()

        System.out.println("\tVector after adding element : "+ v);

        System.out.println("\n2.capacity()");

        System.out.println("\tVector capacity: "+v.capacity());

        //3.clear()

        System.out.println("\n3.clear()");

        v.clear();

        System.out.println("\tVector after using clear() function : "+ v);

        v.add(1);

        v.add(16);

        v.add(4651);

        v.add(41);

        //4.contains()

        System.out.println("\n4.contains()");

        System.out.println("\tvector :"+ v);

        if(v.contains(41)) System.out.println("\tvector contains element 41");

        else System.out.println("\tvector does not contains element 41");

        //5.indexOf()

        System.out.println("\n5.indexOf()");

        System.out.println("\tIndex of element 16: "+ v.indexOf(16));

        //6.insertElementAt()

        System.out.println("\n 6.insertElementAt()");

        System.out.println("\tvector before inserting element:"+ v);

        v.insertElementAt(26, 2);

        System.out.println("\tVector after inserting element at index 2:"+ v);

        //7.elementAt()

        System.out.println("\n7.elementAt()");

        System.out.println("\telement at index 3 :"+v.elementAt(3));

        //8.isEmpty()

        System.out.println("\n8.isEmpty()");

        if(v.isEmpty()) System.out.println("\tvector is Empty");

        else System.out.println("\tvector is not empty");

        //9.remove()

        System.out.println("\n9.remove()");

        System.out.println("\tvector before deleting element:"+ v);

        v.remove(1);

        System.out.println("\tVector after deleting element at index 1:"+v);

        v.add(12);

        //10. sort()

        System.out.println("\n10.sort()");

        System.out.println("\tvector before sortingt:"+ v);

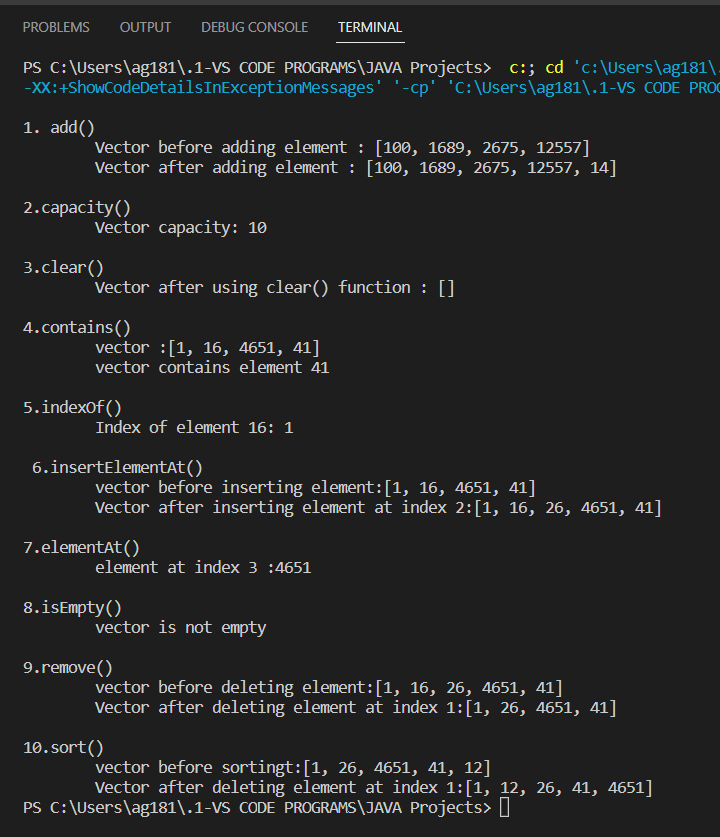
        Collections.sort(v);

        System.out.println("\tVector after deleting element at index 1:"+v);

        }

}

**OUTPUT:**

****